We Claim:

1. A method for determining the possibility of adverse effect arising from a code change in a computer program, comprising the steps of:

identifying important classes within a computer program;

determining directly and indirectly dependent classes of said important classes;

associating test cases with said important classes and with said directly and dependent classes; and

for a given code change to an important class:

running all said associated test cases; and

indicating the possibility of an adverse effect if any run test case fails.

2. The method of claim 1, wherein the identification of important classes includes building an inheritance structure of class names and super classes of said program, and from which said structure start points and direct and indirect descendants thereof are identified.

3. The method of claim 2, wherein determining dependent classes includes:

finding references in said program to said important classes;

finding methods invoked by said important classes; and

determining a dependency structure of said methods that incorporates said dependent classes.

4. The method of claim 3, wherein determining said dependency structure further includes identifying both directly dependent and indirectly dependent classes, said indirectly dependent classes exhibiting a producer/consumer relation for persistent data.

5. The method of claim 1, wherein indicating an adverse step includes generating a program output.

6. A method for determining the possibility of adverse effect arising from a code change in a computer program having a plurality of classes, comprising the steps of:

identifying important ones of said classes;

determining directly and indirectly dependent classes of said important classes;

associating test cases with said classes and with said directly and dependent classes; and

for a given code change to a class:

running all said associated test cases; and

indicating the possibility of an adverse effect if any run test case fails.

7. The method of claim 6, wherein the identification of important classes includes building an inheritance structure of class names and super classes of said program, and from which said structure start points and direct and indirect descendants thereof are identified.

8. The method of claim 7, wherein determining dependent classes includes:

finding references in said program to said important classes;

finding methods invoked by said important classes; and

determining a dependency structure of said methods that incorporates said dependent classes.

9. The method of claim 8, wherein determining said dependency structure further includes identifying both directly dependent and indirectly dependent classes, said indirectly dependent classes exhibiting a producer/consumer relation for persistent data.

10. The method of claim 6, wherein indicating an adverse step includes generating a program output.

11. A data processing system comprising:

a memory storing a program, the program having a plurality of classes;

a user input by which program code changes can be made;

a processor operable to identify important ones of said classes, determine directly and indirectly dependent classes of said important classes, and associate test cases with said classes and with said directly and dependent classes; and

wherein for a given code change to a class nput via said user input, said processor runs all said associated test cases; and

further comprising:

an output means by which a program output is generated to indicate the possibility of an adverse effect if any run test case fails.

12. The data processing system of claim 11, wherein said processor is further operable to identify important classes by building an inheritance structure of class names and super classes of said program, and from which said structure start points and direct and indirect descendants thereof are identified.

13. The data processing system of claim 12, wherein said processor is further operable to determine dependent classes by finding references in said program to said important classes, finding methods invoked by said important classes, and determining a dependency structure of said methods that incorporates said dependent classes.

14. The data processing system of claim 13, wherein said processor is further operable to determine said dependency structure by identifying both directly dependent and indirectly dependent classes, said indirectly dependent classes exhibiting a producer/consumer relation for persistent data.

15. A computer program product comprising a computer program carried on a storgae medium, said computer program comprising:

a program code element having a plurality of classes;

a code element for identifying important ones of said classes;

a code element for determining directly and indirectly dependent classes of said important classes;

a code element for associating test cases with said classes and with said directly and dependent classes;

a code element for running all said associated test cases for a class contingent upon a given code change to said class within said program code element; and

a code element for indicating the possibility of an adverse effect if any run test case fails.

16. A computer program product comprising computer program carried on a storgae medium, said computer program comprising:

a program code element having a plurality of classes;

a code element for identifying important classes within a computer program;

a code element for determining directly and indirectly dependent classes of said important classes;

a code element for associating test cases with said important classes and with said directly and dependent classes;

a code element for running all said associated test cases for a class contingent upon a given code change to said class within said program code element; and

a code element for indicating the possibility of an adverse effect if any run test case fails.